Toward human-in-the-loop PID control based on CACLA reinforcement learning

Junpei Zhong[1] and Yanan Li[2]

[1] Nottingham Trent University, Nottingham, NG11 8NS, United Kingdom
zhong@junpei.eu,
[2] University of Sussex, Falmer, Sussex, BN1 9RH, United Kingdom

Abstract. A self-tuning PID control strategy using a reinforcement learning method, called CACLA (Continuous Actor-critic Learning Automata) is proposed in this paper with the example application of human-in-the-loop physical assistive control. An advantage of using reinforcement learning is that it can be done in an online manner. Moreover, since human is a time-variant system. The demonstration also shows that the reinforcement learning framework would be beneficial to give semi-supervision signal to reinforce the positive learning performance in any time-step.

Keywords: human-in-the-loop, reinforcement learning, adaptive control

1 Introduction

As a result of the recent development of intelligent robotics, it can be seen that there are more scenarios that robotic application will have the following trends:

1. it will have physical or social interact with humans in industries, health-care and domestic uses;
2. more and more sensory information becomes available due to the rapid development and deployment of sensor hardware.

Both of above two trends can also become an overlapped question, which is: How we can develop adaptive method to have a better control policy of the robots, so that we can learn the profile of the individual difference of the users.

For instance, as one the classical control methods to achieve effective performance in matching the control criteria, the PID controllers have been used in various kinds robotic controllers, because of its structural simplicity and acceptable performances. Most of the conventional methods use the method of trial and error to tune the optimal PID constants. For instance, the first tuning rule of PID was proposed in [1]. Moreover, there are several tuning rules have been proposed. (see also [2]). Some knowledge learning methods have been proposed to adapt the uncertainties of the control plant, in the way to represent the continuous updating of the parameters as the knowledge. For instance, the fuzzy

rules are used to mimic the human tuning procedures. In [3], a hybrid control system incoporating a fuzzy controller a PI controller in the steady-state is proposed. [4] suggested a fuzzy rule-based tuning method to adjust the PI gains. But both of the above methods cannot applied in such time-variant system, since the control rules and membership functions of fuzzy controller are hard-coded [5].

A few learning systems have also been developed to solve the parameter optimization problem in the time-variant systems by setting the system parameters based on neural network or fuzzy systems [6]. However, one assumption of using such learning methods is that the search space is continuous and differentially smooth. Thus for a complicated system, these methods usually have a low convergence speed. Although some non-linear search algorithms, such as the genetic algorithm (GA)[7], simulated annealing (SA)[8] and evolutionary programming (EP) [9, 10] have been proposed for searching in a non-convex space, these methods are also difficult to apply in the human-in-the-loop learning settings, because most of them lack the ability to do on-line optimization.

Since the presence of human has been one of the factors in the control system, human-in-the-loop learning provide an effective way of obtaining the optimal values for the time-variant systems. By taking into account the human as part of the system itself, at one hand the human can act as a variable of the system. On the other hand, human can be involved in training, tuning and testing the data by providing feedbacks into the results. Therefore, obtaining learning robot skills by human-in-the-loop learning can be more efficient. By incorporating the human in the multi-agent system, the control and deployment of unmanned aerial vehicles (UAVs) show improvement [11]. [12] uses EMG signal as a measure of the human feedback to control exoskeleton robots. Particularly, reinforcement learning, as one of the semi-supervised learning method, is a friendly and straight-ward approach for the non-experienced users to give feedback for the robots without the explicit human demonstration. For instance, in the case of physical human-robot interaction (pHRI), it can be represented as a system that incorprates two sub-systems: the human and the robot. And we could simply these sub-systems as the same model: a mass-damper-spring system [13]. Nevertheless, the robotic sub-system should also learn the parameters of the human-mass-damper-spring system as a time-variant system, which will be the focus of our following research.

Thus, the purpose of this article is to present a solution of the human-in-the-loop system using an RL algorithm, aiming at two main requirements :

1. incorporate the knowledge added by the users to allow the agent to develop progressively its own knowledge;
2. dealing with continuous state and updates of the adaptabilities in an on-line manner.

2 Reinforcement Learning

In this paper, the reinforcement learning (RL) [14] is used to adaptively change the parameters of PID controller with part of the knowledge from the users and

the environment, which is called semi-supervised learning. The general idea of RL is that the environment can always evaluate the exploitation results while the RL method is trying to exploit the environment. Different from the learning method unlike supervised learning, which the correct results will be given, or the "trial and error" that not any feedbacks are given.

2.1 Continuous Actor-Critic Automaton

Reinforcement learning (RL) is a framework for solving sequential decision problems, in which the model learns to take better decisions, which are called "actions" which represent the transit between each states while interacting with its environment. Such kinds of actions can be explicitly modelled as policies, or just a scalar value. Once the model performs an action, the state changes and the agent receives another values regarding to the current state. The underlying formalism of RL is that of Markov Decision Processes (MDP). An MDP is formally defined as a tuple, states S, actions A, models/transition models T, and rewards R. And the traditional MDP formulation usually employs the discrete grid world as the representation of both the state and actions.

In the case of adaptive PID control, for instance, the state can represent the combination of the parameters. Furthermore, there are values corresponding to each state. This value is called reward, which encodes information about the quality of the actual transition. The goal of the agent is to maximize the long-term expected total reward, which is equivalent to finding the optimal solution of an MDP.

In the MDP formulation, the set of actions T are defined as

$$T : S \times A \times S \to [0, 1] \tag{1}$$

where T determines the transition probabilities between states ($T(s, a, s_0) = p(s_0|a, s)$ is the probability from the current state s_0 from the next state s while the action a is executed. Then the reward signal is updated as

$$R : S \times A \to R \tag{2}$$

is a reward signal. It represents the signal that after transitioning from state S with an action A.

A policy, encoding how the model will behave is defined as

$$\pi : S \times A \to [0, 1] \tag{3}$$

Usually when using π, we always try to maximize the expected discounted reward:

$$\pi^* = \arg\max_{\pi} J(x) \tag{4}$$

$$= \arg\max_{\pi} E[\sum_{t=0}^{\infty} \gamma^t \times R[s_t, \pi_t(s_t)]] \tag{5}$$

where t denotes the time-steps and $0 \in (0, 1)$ is a discount factor.

By doing so, the Value V correspond to each state is also imported which represents the expected sum of rewards accumulated starting from state s, acting optimally the reach the final reward:

$$V^*_{t+1}(s) \leftarrow max \sum_{S=s}^{\infty}(s, a, s)[R(s, a, s) + \gamma V^*_i(s')] \tag{6}$$

The continous actor-critic automata (CACLA) [15] follows the same principle of the MDP and general RL methods, but acts in the continuous space in both state and action representations. One significant change in algorithm while changing from discrete to continuous space is that the Values corresponding to each state are only updated while the temporal error γ is positive, which suggests that the latest performed (explorable) action leads to a larger reward than expected :

$$\delta_t = r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t) \tag{7}$$

The CACLA is often implemented in neural networks as the universal approximators, for approximating both the critic and the actor values in the continuous domains.

3 Case Study: Robot-assisted Recovery

3.1 Problem Formulation

In this paper, without loss of generality, we take a simplified example of using a assisted physical training robot for the upper-limb to test the CACLA based human-in-the-loop PID control. In this case, as shown in Fig. 1, which is a rehabilitation robot for the upper limb. To use it, the a human arm is holding to the robot while the robot is passively adapting the driven power to assist the human to recover the upper limb.

In this case, the dynamics of a planar physical training robot are given by

$$M_r(x)\,\ddot{x} + C_r(x, \dot{x}) \tag{8}$$

$$\dot{x} = u + f \tag{9}$$

where x is the position in the task space, $M_r(q)$ is the robot's inertia/mass matrix, $C_r(x, \dot{x})$ is the robot's Coriolis and centrifugal matrix, u is the robot's control input from its motors and f is the force applied by the human.

At the meanwhile, the dynamics of a human arm are given by

$$M_h(x)\,\ddot{x} + C_h(x, \dot{x}) \tag{10}$$

$$\dot{x} = u_h - f \tag{11}$$

where $M_h(x)$ is the human arm's inertia/mass matrix, $C_h(x, \dot{x})$ is the human arm's Coriolis and centrifugal matrix and u_h is the human's control input. Note

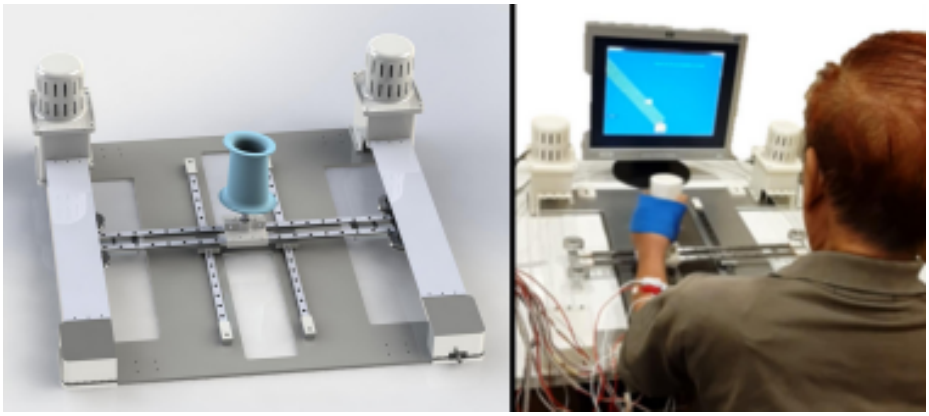


Fig. 1: An example of a rehabilitation robot for the upper limb, while the force of the human user should be taken into account. Adopted from [16]

that x is the common position in Eqs. 9 and 11 which is the position of the interaction point of the robot and human arm.

While the human and the robot are interacting together, we combine the Eqs. 9 and 11, and thus the dynamics of the combined system are as below

$$M(x)\ddot{x} + C(x, \dot{x}) \tag{12}$$

$$\dot{x} = u + u_h \tag{13}$$

where the interaction force f disappears and

$$M(x) = M_r + M_h(x) \tag{14}$$

$$C(x, \dot{x}) = C_r(x, \dot{x}) + C_h(x, \dot{x}) \tag{15}$$

Therefore, the combined dynamics are determined by both the robot's and human's control inputs u and u_h. In this case, we need to the control policy to satisfy with two conditions:

- to track with trajectory with the desired positions with minimum errors;
- to fit with the desired requirement for the power executed by the human user, i.e. the human's control input u_h.

For this purpose, the human's unknown control input can be constructed based on certain periodic parameters, as follows

$$u_h = -K_{h1}(x - x_d) - K_{h2}\dot{x} \tag{16}$$

where K_{h1} and K_{h2} are the human's stiffness and damping parameters, respectively and x_d is the desired trajectory that is defined for a task. K_{h1} and K_{h2} are unknown parameters that can be time-varying during a tracking task.

3.2 Adaptive Parameters using CACLA

In the scenario of PD control, it is difficult to get the optimal combination of parameters. Especially, when the manipulation is done by the collaboration between human and robot, it is necessary to adapt the individual difference between users or even the changes of impedance control paradigms.

The K_p and K_d are represented as a 2-dimensional input state space X in the CACLA network. The size of the state space is defined as $I_a \times I_b$, where the I indicates the index of the 2-dimensional state space. The neural activation $x_{a,b}$ state space is determined by the distance between the particular neuron and the represented parameters K_p and K_d in a Gaussian form:

$$x_{i_a,i_b} = e^{\frac{-distance}{2\sigma^2}} \tag{17}$$

where the distance is defined as

$$distance = \sqrt{(I_a - K_p)^2 + (I_b - K_d)^2} \tag{18}$$

Besides of the input, the CACLA network include two output units: the action and the critic. They are connected with:

$$action = XW_{act} \tag{19}$$

$$critic = XW_{cri} \tag{20}$$

where the X is the state space. the output of the action unit range from $[0, 2\pi)$, and the critic is usually below 1.

The update of the input space can be regarded as a movement of the combination of K_p and K_d. The movement is with a constant radius 1 but with different degree defined by the action output.

3.3 Case study

A sine curve is chosen as the desired trajectory for the movement:

$$\hat{x} = 0.3 \times sin(t) \tag{21}$$

Referring to Eq.9:

$$M = 5; \tag{22}$$

$$C = 0.1 \tag{23}$$

For the adaptive learning, iterations are employed to learn the optimal parameters online. In each iteration, a state of (K_p, K_d) is randomly selected in the state space. Then a update is made according to the output of action unit: it can be illustrated as a constant movement with the radius of 0.1 in the state space

but with various degree of angle, which is the output of the action unit *action*. The iteration terminates while it reaches the pre-defined maximum number of iterations, or the signal of tracking the desired trajectory (with the pre-defined minimum error) is detected.

The detailed algorithm is as Algorithm 1:

Algorithm 1 CACLA Training

procedure ONE ITERATION(*withrandominitialspace*)
  while *error* > *threshold* or *iteration* > *maximum_iteration* **do**
    ▷ Repeat iteration for one sequence until threshold is achieved
    Output *action* and *critic*. (Eq.20)
    W_{act} and W_{cri} updated
    if $\delta > 0$ (Eq. 7) **then**
      Agent moves
    end if
  end while
end procedure

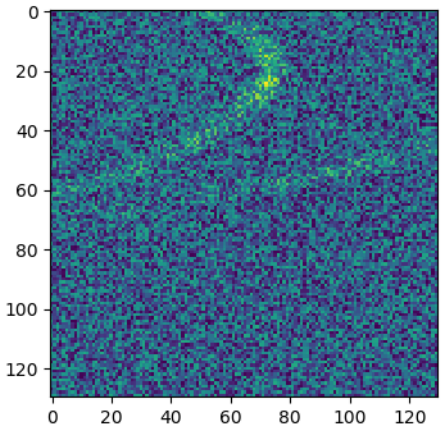
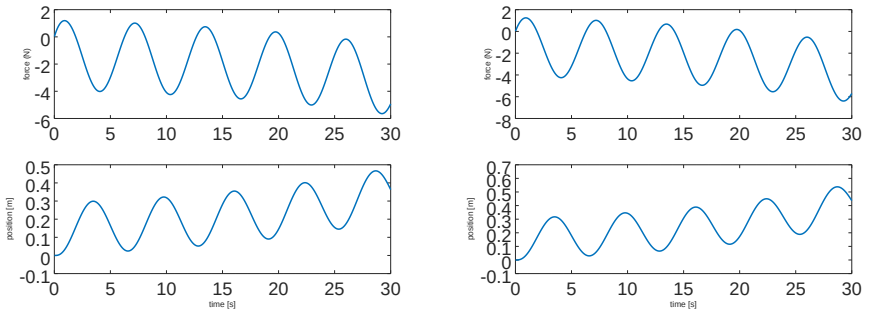


Fig. 2: Weighting matrix of the critic value. The optimal values of K_p and K_d can be obtained by the values on the axes divide by 10.

Figure 2 demonstrates the resulting values of the K_p and K_d. The green trace in the middle of the weighting matrix indicates the optimal values for K_p and K_d are around $(K_p = 8, K_d = 2)$. According to this, we select the values to see the performance of the system. As we can see in Fig. 3a, the trajectory of the robot basically follows the desired one. For comparison, we also select other combinations along the yellow trace of the weighting matrix. From Fig. 3b, a

satisfying result can be also seen. More importantly, thanks to the continuous values of the weighting, we do not need to start training with every possible combination of the parameters, which is beneficial for the convex optimization problem with limited training samples from human.



(a) $K_p = 8, K_d = 2$ (b) $K_p = 10, K_d = 6$

Fig. 3: The trajectories and forces with different parameter settings

4 Summary

In this paper, we propose to utilise a continuous reinforcement learning algorithm, named in both state and action space, to search for an optimal setting of the PID control system. Particularly, the system learns the parameters in an on-line time-variant manner, which is suitable to incorporate human-factors in the reinforcement learning loop. We also examine this learning mechanism in a 1-dimensional physical-assistive robot which include both human and robotic systems. The initial result shows that the adaptive PD control with this system can track the desired trajectory. At the next steps, the learning algorithm will be further elaborated to solve more complicated human-in-the-loop control systems.

References

1. J. G. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers," *trans. ASME*, vol. 64, no. 11, 1942.
2. K. J. Åström, T. Hägglund, and K. J. Astrom, *Advanced PID control*. ISA-The Instrumentation, Systems, and Automation Society Research Triangle , 2006, vol. 461.
3. A. V. Sant and K. Rajagopal, "Pm synchronous motor speed control using hybrid fuzzy-pi with novel switching functions," *IEEE Transactions on Magnetics*, vol. 45, no. 10, pp. 4672–4675, 2009.
4. J.-W. Jung, Y.-S. Choi, V. Leu, and H. Choi, "Fuzzy pi-type current controllers for permanent magnet synchronous motors," *IET electric power applications*, vol. 5, no. 1, pp. 143–152, 2011.
5. Y. Li, K. H. Ang, and G. C. Chong, "Pid control system analysis and design," *IEEE Control Systems Magazine*, vol. 26, no. 1, pp. 32–41, 2006.
6. G. Orelind, L. Wozniak, J. Medanic, and T. Whittemore, "Optimal pid gain schedule for hydrogenerators-design and application," *IEEE Transactions on energy Conversion*, vol. 4, no. 3, pp. 300–307, 1989.
7. B. Porter and A. Jones, "Genetic tuning of digital pid controllers," *Electronics letters*, vol. 28, no. 9, pp. 843–844, 1992.
8. C. Roa-Sepulveda and B. Pavez-Lazo, "A solution to the optimal power flow using simulated annealing," *International journal of electrical power & energy systems*, vol. 25, no. 1, pp. 47–57, 2003.
9. Á. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Transactions on evolutionary computation*, vol. 3, no. 2, pp. 124–141, 1999.
10. J. Lieslehto, "Pid controller tuning using evolutionary programming," in *Proceedings of the 2001 American Control Conference.(Cat. No. 01CH37148)*, vol. 4. IEEE, 2001, pp. 2828–2833.
11. M. Orsag, T. Haus, D. Tolić, A. Ivanovic, M. Car, I. Palunko, and S. Bogdan, "Human-in-the-loop control of multi-agent aerial systems," in *2016 European Control Conference (ECC)*. IEEE, 2016, pp. 2139–2145.
12. L. Peternel, T. Noda, T. Petrič, A. Ude, J. Morimoto, and J. Babič, "Adaptive control of exoskeleton robots for periodic assistive behaviours based on emg feedback minimisation," *PloS one*, vol. 11, no. 2, p. e0148942, 2016.
13. A. Albu-Schäffer, C. Ott, and G. Hirzinger, "A unified passivity-based control framework for position, torque and impedance control of flexible joint robots," *The international journal of robotics research*, vol. 26, no. 1, pp. 23–39, 2007.
14. R. S. Sutton, A. G. Barto *et al.*, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 135.
15. H. van Hasselt and M. Wiering, "Reinforcement learning in continuous action spaces," in *IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning, 2007.*, 2007, pp. 272–279.
16. A. Hussain, A. Budhota, C. M. L. Hughes, W. D. Dailey, D. A. Vishwanath, C. W. Kuah, L. H. Yam, Y. J. Loh, L. Xiang, K. S. Chua *et al.*, "Self-paced reaching after stroke: A quantitative assessment of longitudinal and directional sensitivity using the h-man planar robot for upper limb neurorehabilitation," *Frontiers in neuroscience*, vol. 10, p. 477, 2016.